Unify DevOps and SecOps: Security Without Friction

Matt Alderman, CISSP
Chief Strategy & Marketing Officer
Layered Insight
@maldermania
Technology Trend #1: Infrastructure Migrates to the Cloud
Technology Trend #2: IoT Accelerates Edge Computing
Technology Trend #3: Container Applications are the Future
The People Challenge

DevOps Benefits...

Agility

Velocity (Production, Market, Improvement, and Innovation)

Optimization (Productivity, Costs, and Competitiveness)

DevOps Challenges...

Not Understanding Security

Automating to Enhance Agility

Starting with Technology Solutions
SecOps Wants...

Inventory Of Authorized and Unauthorized Software
Compliance with Policies and Regulations
Security Assessment, Monitoring, and Remediation

SecOps Challenges...

Adjusting to Speed
Security’s Changing Role
Legacy Solutions Limit Visibility and Control
The People Challenge

DevOps

SPEED?
AGILITY?
PORTABILITY?

SecOps

INVENTORY?
COMPLIANCE?
SECURITY?
The People Solution: Unify DevOps and SecOps

Unify DevOps and SecOps to provide complete visibility and control
The Technology Challenge

2016 Challenges

1. Shrinking infrastructure, as organizations continue migration to the cloud
2. Containers deployed within Virtual Machines
3. But organizations still have the overhead and costs of the hypervisor and virtual machines
2017 Challenges

1. The orchestration battle ends with Kubernetes winning 80% of the market
2. But organizations struggle to scale their own Kubernetes clusters
3. Also, where do you put security without creating dependencies?
2018 Challenges

1. Container-as-a-Service and Orchestration-as-a-Service adoption accelerate container adoption
2. Now where do you put security?
The Three Approaches

Approach #1: Kernel Plugin with Root Access

Approach #2: Privileged Container with Root Access

Approach #3: Container Native with User Access
The Three Approaches

#1: Kernel Plugin

- Challenges
  - Has Root Privileges
  - Lacks App Visibility
  - Limits Portability
The Three Approaches

#1: Kernel Plugin
- Container
- Container
- Container
- Container Engine
- Operating System
- Kernel
- Infrastructure

Challenges
- Has Root Privileges
- Lacks App Visibility
- Limits Portability

#2: Privileged Container
- Container
- Privilege
- Container
- Container Engine
- Operating System
- Kernel
- Infrastructure

Challenges
- Requires Root Access
- Lacks Scalability
- Limits Portability
The Three Approaches

#1: Kernel Plugin
- Container
- Container
- Container
- Container Engine
- Operating System
- Kernel
- Infrastructure

Challenges
- Has Root Privileges
- Lacks App Visibility
- Limits Portability

Advantages
- No Root Privileges
- Zero-Touch

#2: Privileged Container
- Container
- Container Engine
- Operating System
- Kernel
- Infrastructure

Challenges
- Requires Root Access
- Lacks Scalability
- Limits Portability

#3: Container Native
- Secured Container
- Secured Container
- Secured Container
- Container Engine
- Operating System
- Kernel
- Infrastructure

Challenges
- Secured Container
- Secured Container
- Secured Container

Advantages
- Scales Elastically
- Infinite Portability
Solution

1. Container Native injects sensors within the container image to monitor application, system, network, and file activity

2. Container Native protects the Containers independently of the underlying infrastructure
The Process Challenge

The diagram illustrates the process of managing container security. It highlights the following steps:

1. **New container build with basic unit test**
2. **Post to registry**
3. **Insecure container sent to Sec**
4. **Sec sends Insecure Container to Dev for re-imaging**

The process begins with a new container build and basic unit test. The container is then posted to the registry. An insecure container is sent to Sec, and Sec sends the insecure container to Dev for re-imaging.
The Process Solution: Security Without Friction

**Build**
- New container build with basic unit test
- git commit, git push
- Dev

**Ship**
- Container Registry
  - Insecure Container posted to registry
  - Copy of Original Container
  - Instrumented Container inserted into registry

**Assessment/Instrumentation**
- Container Registry
  - Static Analysis
    - Software Inventory
    - Security Assessment
  - Runtime Analysis
    - Compliance Enforcement
    - Pass/Fail
  - Runtime Protection
    - Machine Learning
    - Security Policies

**Test/Staging**
- Insecure Container
  - Pull container image from registry
- Instrumented Container
  - Instrumented Container inserted into registry
  - Instrumented Container runs in observation mode on premise or in cloud

**Run**
- DevOps
- SecOps
- Automated enforcement of Secured Container on premise or in cloud
- Insecure Container
  - Posted to registry
  - Insecure Container
- New container build with basic unit test
- Any failed containers are returned to DevOps

**Production**
- Failed Container
  - Copy of Original Container
  - Insecure Container
  - New Container
  - New Container
  - << FAIL PASS >>
Conclusion & Call To Action

People: Unify DevOps and SecOps

1. Do you have a formal DevOps team?
2. Are they developing with containers?
3. Are they deploying or preparing to deploy containers?
Container Native Application Protection

1. Are you deploying on-premises or in the cloud?

2. Are you looking at CaaS/serverless infrastructure?

3. Do you want to future-proof security?
Process: Security Without Friction

1. What is the current CI/CD process?
2. How do you embed security within the process?
3. How do you integrate with your exiting SecOps/SOC process?
Thank You